Solving the Independent Set Problem by Using Tissue-Like P Systems with Cell Division

Mario J. Pérez-Jiménez[2], and Agustín Riscos-Núñez[2]

[1] Research Group on Computational Topology and Applied Mathematics
sbdani@us.es
[2] Research Group on Natural Computing
magutier@us.es, ariscosn@us.es, marper@us.es
University of Sevilla
Avda. Reina Mercedes s/n, 41012, Sevilla, Spain

Abstract. Tissue-like P systems with cell division is a computing model in the framework of Membrane Computing inspired by the intercellular communication and neuronal synaptics. It considers the cells as unit processors and the computation is performed by the parallel application of given rules. Division rules allow an increase of the number of cells during the computation. We present a polynomial-time solution for the Independent Set problem via a uniform family of such systems.

1 Introduction

In the last years, Membrane Computing is becoming one of the pillars of Natural Computing. In the similar way as other research fields in Natural Computing, as *Genetic Algorithms*, *Neural Networks*, or *DNA Computing*, it takes advantage from the way in which Nature computes.

Membrane Computing is a theoretical model of computation inspired by the structure and functioning of cells as living organisms able to process and generate information. The computational devices in Membrane Computing are called *P systems*. Roughly speaking, a P system consists of a membrane structure, in the compartments of which one places multisets of objects which evolve according to given rules. In the most extended model, the rules are applied in a synchronous non-deterministic maximally parallel manner, but some other semantics are being explored (see [14] for details).

Since the seminal paper [10], different models of P systems have been studied. According to their architecture, these models can be split into two sets: cell-like P systems and tissue-like P systems. This paper is devoted to the second approach: tissue-like P systems. According to the architecture, the main difference with cell-like P systems is that the structure of membranes is defined by a general graph instead of a tree-like graph. This kind of models was first presented by Martín–Vide *et al.* in [7] and it has two biological inspirations (see [8]): intercellular communication and cooperation between neurons. The communication among cells is based on symport/antiport rules [12]. Symport rules move objects across a

membrane together in one direction, whereas antiport rules move objects across a membrane in opposite directions.

The search of the abilities of tissue-like P systems as computational model has leaded many authors to consider new models with small differences from the original one (see, for example, [4,5,6]) . One of the most interesting variants of tissue P systems was presented in [13] (and it was studied in depth in [1]). In that paper, tissue P systems are endowed with the ability of getting new cells based on cellular division, yielding *tissue-like P systems with cell division*.

This is a further step in the analogy between a theoretical computational model and living tissues. In a living tissue, new cells are obtained via cellular division or *mitosis*. This ability allows the tissue to grow till reaching its maturity or repairing damages. Cells divide themselves till reaching a number of copies big enough to perform its activity successfully. Following such analogy, tissue-like P systems with cell division are endowed with the ability of obtaining a number of cells greater than the original by means of a rule which implement the idea of mitosis. The computational meaning is that we can create an exponential number of cells in linear time. Each individual cell can be seen as a processor and the cells work in parallel.

The ability of obtaining an exponential amount of workspace in polynomial-time has been the basis to open a new landscape for the study of a key problem in the theory of computational complexity: **P** vs. **NP**. Finding biologically inspired differences among the different P system models that may determine different borderlines between tractability and intractability is the key problem of the Complexity Theory in P systems. Some **NP**-complete problems have been efficiently solved with tissue-like P systems with cell division: SAT [13], 3–coloring [2] or Subset Sum [3]. In this paper, we extend our study by presenting a polynomial–time solution to the Independent Set problem.

The paper is organized as follows: in Section 2 we recall the definition of Tissue-like P systems with cell division. A linear–time solution to the Independent Set problem with the necessary resources and the main results are presented in the following section. Section 4 includes a short overview of the computations. Finally, some conclusions and new open research lines are presented.

2 Tissue-Like P Systems with Cell Division

In the first definition of the model of tissue P systems [7,8] the membrane structure did not change along the computation. Based on the cell-like model of P systems with active membranes, Gh. Păun et al. presented in [13] a new model of tissue P systems *with cell division*. The biological inspiration is clear: alive tissues are not *static* network of cells, since cells are duplicated via mitosis in a natural way.

The main features of this model, from the computational point of view, are that cells have not polarizations (the contrary holds in the cell-like model of P systems with active membranes, see [11]); the cells obtained by division have the same labels as the original cell; and if a cell is divided, its interaction with other cells or with the environment is blocked during the mitosis process.

Formally, a *tissue-like P system with cell division* of degree $q \geq 1$ is a tuple of the form $\Pi = (\Gamma, \mathcal{E}, w_1, \ldots, w_q, \mathcal{R}, i_0)$, where:

1. Γ is a finite *alphabet* (a set of symbols that will be called *objects*).
2. $\mathcal{E} \subseteq \Gamma$ (the objects in the environment).
3. $w_1, \ldots, w_q \in \Gamma^*$ are strings over Γ representing the multisets of objects associated with the cells at the initial configuration.
4. $\mathcal{R}$ is a finite set of rules of the following form:
   (a) *Communication rules*: $(i, u/v, j)$, for $i, j \in \{0, 1, \ldots, q\}, i \neq j$, $u, v \in \Gamma^*$.
   (b) *Division rules*: $[a]_i \rightarrow [b]_i[c]_i$, where $i \in \{1, 2, \ldots, q\}$ and $a, b, c \in \Gamma$.
5. $i_0 \in \{0, 1, 2, \ldots, q\}$.

A tissue-like P system with cell division of degree $q \geq 1$ can be seen as a set of q cells (each one consisting of an elementary membrane) labelled by $1, 2, \ldots, q$. We will use 0 to refer to the label of the environment, and i_0 denotes the output region (which can be the region inside a cell or the environment).

The communication rules determine a virtual graph, where the nodes are the cells and the edges indicate if it is possible for pairs of cells to communicate directly. This is a dynamical graph, because new nodes can be produced by the application of division rules.

The strings $w_1, \ldots, w_q$ describe the multisets of objects placed in the q cells of the system. We interpret that $\mathcal{E} \subseteq \Gamma$ is the set of objects placed in the environment, each one of them available in an arbitrary large amount of copies.

The communication rule $(i, u/v, j)$ can be applied over two cells labelled by i and j such that u is contained in cell i and v is contained in cell j. The application of this rule means that the objects of the multisets represented by u and v are interchanged between the two cells. Note that if either $i = 0$ or $j = 0$ then the objects are interchanged between a cell and the environment.

The division rule $[a]_i \rightarrow [b]_i[c]_i$ is applied over a cell i containing object a. The application of this rule divides this cell into two new cells with the same label. All the objects in the original cell are replicated and copied in each of the new cells, with the exception of the object a, which is replaced by the object b in the first one and by c in the other one.

Rules are used as usual in the framework of membrane computing, that is, in a maximally parallel way (a universal clock is considered). In one step, each object in a membrane can only be used for one rule (non-deterministically chosen when there are several possibilities), but any object which can participate in a rule of any form must do it, i.e, in each step we apply a maximal set of rules. This way of applying rules has only one restriction: when a cell is divided, the division rule is the only one which is applied for that cell in that step.

2.1 Recognizer Tissue-Like P Systems with Cell Division

NP-completeness has been usually studied in the framework of *decision problems*. Let us recall that a decision problem is a pair (I_X, θ_X) where I_X is a language over a finite alphabet (whose elements are called *instances*) and θ_X is a total boolean function over I_X.

In order to study the computing efficiency for solving **NP**-complete decision problems, a special class of tissue P systems with cell division is introduced in [13]: *recognizer tissue-like P systems*. The key idea of such recognizer systems is the same one as from recognizer cell-like P systems.

Recognizer (cell-like) P systems were introduced in [9] and they provide a natural framework to study and solve decision problems within Membrane Computing, since deciding whether an instance of a given problem has an affirmative or negative answer is equivalent to deciding whether a string belongs or not to the language associated with the problem.

In the literature, recognizer (cell-like) P systems are associated with P systems with *input* in a natural way. The data encoding an instance of the decision problem has to be provided to the P system in order to compute the appropriate answer. This is done by codifying each instance as a multiset to be placed in an *input membrane*. The output of the computation (`yes` or `no`) is sent to the environment, and in the last step of the computation.

A recognizer tissue-like P system with cell division of degree $q \geq 1$ is a tuple $\Pi = (\Gamma, \Sigma, \mathcal{E}, w_1, \ldots, w_q, \mathcal{R}, i_{in}, i_0)$, where

- $(\Gamma, \mathcal{E}, w_1, \ldots, w_q, \mathcal{R}, i_0)$ is a tissue-like P system with cell division of degree $q \geq 1$ (as defined in the previous section), $i_0 = env$ and $w_1, \ldots, w_q$ strings over $\Gamma \setminus \Sigma$.
- The working alphabet Γ has two distinguished objects `yes` and `no`, present in some initial multiset w_i, but not present in $\mathcal{E}$.
- Σ is an (input) alphabet strictly contained in Γ.
- $i_{in} \in \{1, \ldots, q\}$ is the input cell.
- All computations halt.
- If $\mathcal{C}$ is a computation of Π, then either the object `yes` or the object `no` (but not both) must have been released into the environment, and only in the last step of the computation.

The computations of the system Π with input $w \in \Sigma^*$ start from a configuration of the form $(w_1, w_2, \ldots, w_{i_{in}} w, \ldots, w_q; \mathcal{E})$, that is, after adding the multiset w to the contents of the input cell i_{in}. We say that $\mathcal{C}$ is an accepting computation (respectively, rejecting computation) if the object `yes` (respectively, `no`) appears in the environment associated to the corresponding halting configuration of $\mathcal{C}$.

Definition 1. *We say that a decision problem $X = (I_X, \theta_X)$ is solvable in polynomial time by a family $\mathbf{\Pi} = \{\Pi(n) : n \in \mathbb{N}\}$ of recognizer tissue-like P systems with cell division if the following holds:*

- *The family $\mathbf{\Pi}$ is* polynomially uniform *by Turing machines, that is, there exists a deterministic Turing machine working in polynomial time which constructs the system $\Pi(n)$ from $n \in \mathbb{N}$.*
- *There exists a pair (cod, s) of polynomial-time computable functions over I_X (called a polynomial encoding from I_X in $\mathbf{\Pi}$) such that:*
  - *for each instance $u \in I_X$, $s(u)$ is a natural number and $cod(u)$ is an input multiset of the system $\Pi(s(u))$;*

- *the family* $\mathbf{\Pi}$ *is* polynomially bounded *with regard to* (X, cod, s)*, that is, there exists a polynomial function* p*, such that for each* $u \in I_X$ *every computation of* $\Pi(s(u))$ *with input* $cod(u)$ *performs at most* $p(|u|)$ *steps;*
- *the family* $\mathbf{\Pi}$ *is* sound *with regard to* (X, cod, s)*, that is, for each* $u \in I_X$*, if there exists an accepting computation of* $\Pi(s(u))$ *with input* $cod(u)$*, then* $\theta_X(u) = 1$*;*
- *the family* $\mathbf{\Pi}$ *is* complete *with regard to* (X, cod, s)*, that is, for each* $u \in I_X$*, if* $\theta_X(u) = 1$*, then every computation of* $\Pi(s(u))$ *with input* $cod(u)$ *is an accepting one.*

In the above definition we have imposed to every system $\Pi(n)$ to be *confluent*, in the following sense: every computation of $\Pi(n)$ with the *same* input multiset must always give the *same* answer.

We denote by $\mathbf{PMC}_{TD}$ the set of all decision problems which can be solved by means of recognizer tissue-like P systems with cell division in polynomial time. This class is closed under polynomial reduction and under complement.

3 A Solution for the Independent Set Problem

Let us recall that an *independent set* of a non-directed graph is a subset of its vertices such that it does not contain any pair of vertices adjacent between them. The number of nodes in the subset is called the size of the independent set.

The `Independent Set Problem (IS)` can be settled as follows: given a non-directed graph, $G = (V, E)$, and a natural number $k \leq |V|$, to determine whether or not G has an independent set of size k.

Next, we will prove that `IS` can be solved in linear time (in the number of nodes and edges of the graph) by a family of recognizer tissue-like P systems with cell division. To this aim, let us construct a family $\mathbf{\Pi} = \{\Pi(n, m, k) : n, m, k \in \mathbb{N}\}$ where each system of the family will process every instance u of the problem given by a graph $G = (V, E)$ with n vertices and m edges, and by a size k of the independent set. More formally, we define $s(u) = \langle n, m, k\rangle = \langle n, \langle m, k\rangle\rangle$, where $\langle x, y\rangle = (x + y)(x + y + 1)/2 + x$ is the Gödel mapping. In order to provide a suitable encoding of this instances into the systems, we will use the objects A_{ij}, with $1 \leq i < j \leq n$, to represent the edges of the graph, and we will provide $cod(u)$ as the initial multiset for the system, where $cod(u)$ is the multiset (A, f) with $A = \{A_{ij} : 1 \leq i < j \leq n\}$ and $f : A \to \mathbb{N}$ such that $f(A_{ij}) = 1$ if $\{v_i, v_j\} \in E$ and $f(A_{ij}) = 0$ if $\{v_i, v_j\} \notin E$. It is easy to check that (cod, s) is a polynomial encoding from I_{IS} in $\mathbf{\Pi}$.

Then, given an instance u of the `IS` problem, the system $\Pi(s(u))$ with input $cod(u)$ decides that instance by a brute force algorithm, implemented in the following four stages:

- *generation stage*: all possible subsets of vertices are generated by applying division rules;
- *pre-checking stage*: only those subsets of size k are selected;

– *checking stage*: we check for each selected subset if they do not contain a pair of adjacent vertices;
– *output stage*: an affirmative or negative answer to the problem is given, according to the results of the previous stage.

The family $\mathbf{\Pi} = \{\Pi(n,m,k) : n,m,k \in \mathbb{N}\}$ of recognizer tissue-like P systems with cell division of degree 2 is defined as follows: for each $n,m,k \in \mathbb{N}, \Pi(n,m,k) = (\Gamma, \Sigma, \mathcal{E}, w_1, w_2, \mathcal{R}, i_{in})$, where

– $\Gamma = \{A_i, B_i, \overline{B}_i, B'_i, P_i : 1 \leq i \leq n\} \cup \{a_i : 1 \leq i \leq 3n+m+\lceil \lg n \rceil + 14\} \cup$
$\{c_i, d_i : 1 \leq i \leq n+1\} \cup \{e_i : 1 \leq i \leq 2n+1\} \cup$
$\{f_i : 0 \leq i \leq n-1\} \cup \{g_i : 1 \leq i \leq \lceil \lg\, n \rceil + 1\} \cup$
$\{l_i : 1 \leq i \leq m + \lceil \lg\, n \rceil + 7\} \cup \{h_i : 1 \leq i \leq \lceil \lg\, m \rceil + 1\} \cup$
$\{B_{ij} : 1 \leq i \leq n \wedge 1 \leq j \leq m\} \cup \{A_{ij}, P_{ij} : 1 \leq i < j \leq n\} \cup$
$\{D_{ij} : 1 \leq i,j \leq n\} \cup \{b, D, F_1, F_2, p, T, S, N, \alpha, \beta, \mathtt{yes}, \mathtt{no}\}$.
– $\Sigma = \{A_{ij} : 1 \leq i < j \leq n\}$.
– $\mathcal{E} = \Gamma \setminus \{a_1, b, c_1, \mathtt{yes}, \mathtt{no}, D, A_1, \ldots, A_n\}$.
– $w_1 = a_1\, b\, c_1\, \mathtt{yes}\, \mathtt{no}$ and $w_2 = D\, A_1 \ldots A_n$.
– $\mathcal{R}$ is the following rules set:
1. *Division rules:*
$r_{1,i} \equiv [A_i]_2 \rightarrow [B_i]_2[\alpha]_2$, for $i = 1, \ldots, n$
2. *Communication rules:*
$r_{2,i} \equiv (1, a_i/a_{i+1}, 0)$, for $i = 1, \ldots, 3n+m+\lceil \lg\, m \rceil + 13$
$r_{3,i} \equiv (1, c_i/c_{i+1}^2, 0)$, for $i = 1, \ldots, n$
$r_4 \equiv (1, c_{n+1}/D, 2)$
$r_5 \equiv (2, c_{n+1}/d_1 e_1, 0)$
$r_{6,i} \equiv (2, e_i/e_{i+1}, 0)$, for $i = 1, \ldots, 2n$
$r_{7,ij} \equiv (2, d_j B_i/D_{ij}, 0)$, for $1 \leq i,j \leq n$
$r_{8,ij} \equiv (2, D_{ij}/\overline{B}_i d_{j+1}, 0)$, for $1 \leq i,j \leq n$
$r_{9,j} \equiv (2, e_{2n+1} d_j/f_{j-1}, 0)$, for $j = 1, \ldots, n$
$r_{10} \equiv (2, f_k/l_1 F_1, 0)$
$r_{11,i} \equiv (2, l_i/l_{i+1}, 0)$, for $i = 1, \ldots, m + \lceil \lg\, n \rceil + 6$
$r_{12} \equiv (2, F_1/pF_2, 0)$
$r_{13} \equiv (2, F_2/g_1 h_1, 0)$
$r_{14,i} \equiv (2, g_i/g_{i+1}^2, 0)$, for $i = 1, \ldots, \lceil \lg\, n \rceil$
$r_{15,i} \equiv (2, h_i/h_{i+1}^2, 0)$, for $i = 1, \ldots, \lceil \lg\, m \rceil$
$r_{16,ij} \equiv (2, \overline{A}_{ij} h_{\lceil \lg\, m \rceil + 1}/P_{ij}, 0)$, for $1 \leq i < j \leq n$
$r_{17,i} \equiv (2, g_{\lceil \lg\, n \rceil + 1} \overline{B}_i/B_{i1}, 0)$, for $i = 1, \ldots, n$
$r_{18,ij} \equiv (2, B_{ij}/B_{ij+1} B'_i, 0)$, for $= 1, \ldots, n$ and $j = 1, \ldots, m$
$r_{19,ij} \equiv (2, P_{ij} B'_i/P_j, 0)$, for $1 \leq i < j \leq n$
$r_{20,j} \equiv (2, P_j B'_j/\beta, 0)$, for $j = 1, \ldots, n$
$r_{21} \equiv (2, p\, \beta/\alpha, 0)$
$r_{22} \equiv (2, l_{m+\lceil \lg\, n \rceil + 7} p/T, 0)$
$r_{23} \equiv (2, T/\alpha, 1)$
$r_{24} \equiv (1, b\, T/S, 0)$
$r_{25} \equiv (1, S\mathtt{yes}/\alpha, 0)$

$r_{26} \equiv (1, a_{2n+m+\lceil \lg\, n\rceil+14}b\,/N, 0)$
$r_{27} \equiv (1, \mathtt{no}N/\alpha, 0)$

- $i_{in} = 2$, is the input cell.

In order to establish that the family $\mathbf{\Pi}$ is polynomially uniform by deterministic Turing machines we firstly note that the sets of rules associated with the systems $\Pi(n, m, k)$ are recursively defined. Hence, it suffices to justify that the amount of necessary resources for defining the systems is polynomial in $\max\{n, m, \lceil \lg k\rceil\}$. In fact, it is polynomial in $\max\{n, m\}$, since those resources are the following:

1. Size of the alphabet: $n(5n-1)/2+n\cdot m+8n+2m+3\lceil \lg\, n\rceil+\lceil \lg\, m\rceil+36 \in \Theta(n^2+m)$.
2. Initial number of cells: $2 \in \Theta(1)$.
3. Initial number of objects: $n+6 \in \Theta(n)$.
4. Number of rules: $3n^2+n\cdot m+9n+2\lceil \lg\, n\rceil+\lceil \lg\, m\rceil+31 \in \Theta(n^2+nm)$.
5. Upper bound for the length of the rules: $3 \in \Theta(1)$

As we will see in the following section, the family $\mathbf{\Pi}$ is also polynomially (in fact, linearly) bounded, sound and complete with regard to $(\mathtt{IS}, cod, s)$. So, we have the main result of the paper.

Theorem 1. $\mathit{IS} \in \mathbf{PMC}_{TD}$

Taking into account that IS is an **NP**–complete problem, and that the class $\mathbf{PMC}_{TD}$ is closed under complement, the following is deduced.

Corollary 1. $\mathbf{NP} \cup \mathbf{co-NP} \subseteq \mathbf{PMC}_{TD}$

4 An Overview of the Computations

Next, we describe in detail the steps followed by the system $\Pi(s(u))$ when the input multiset $cod(u)$ is supplied, for an arbitrary instance u of IS. Let us note that the system starts with only two cells, one labelled by 1 and the other labelled by 2, and that division rules are only applied to cells labelled by 2. This means that along the computations there will always be a unique cell labelled by 1 (which we will call the 1-cell), but that new cells labelled by 2 (which we will call the 2-cells) will be produced.

In order for the system $\Pi(s(u))$ to decide the instance of the IS problem encoded by $cod(u)$, it starts with the *generation stage*, where cells for all the possible subsets of nodes of the graph are generated. This is performed by the successive application of the division rules. These rules take the objects A_i in the 2-cells, which encode the vertices of the graph, and produce two new 2-cells, one of them with the object B_i, meaning that we include the vertex in the subset, and the other without it, meaning that we do not include the vertex in the subset. Of course, all the remaining objects contained in the original 2-cells are replicated into the new ones. This way, at the end of this stage, which spends n steps, the system will have 2^n 2-cells, each of them encoding one and only one subset of vertices of the graph, by means of the objects B_i.

To control the end of this stage, the objects c_i in the 1-cell of the system are used as counters. Initially object c_1 is interchanged for two objects c_2 from the environment using rule $r_{3,1}$; each of these objects are again interchanged for two objects c_3 using rule $r_{3,2}$; and so on. Thus, at the end of the generation stage the 1-cell will contain 2^{n+1} objects c_{n+1}. On the other hand, objects a_i in the 1-cell of the system are used as global counters of the computation by means of the rules $r_{2,i}$. Note that this generation stage is non-deterministic, but it is easy to check its confluence: independently of the way the division rules are applied, at the end of the stage the same configuration is always reached.

Once all the subsets of vertices of the graph are generated, the *pre-checking stage* selects only those of size k. This stage is activated by the rules r_4 and r_5, which interchange the object D of each 2-cell (recall that there are 2^n of them) with an object c_{n+1} of the 1-cell (recall that there are 2^{n+1} of them), and then each of the latter in each 2-cell with one object d_1 and one object e_1 of the environment (recall that there are infinitely many of them). From now on, the 1-cell will wait counting the number of steps of the computation by means the objects a_i and the rules $r_{2,i}$.

The objects d_1 and e_1 start two processes of counting in each 2-cell. The first one counts the number of steps of the stage that have been performed, and it is controlled by the objects e_i, which are repeatedly interchanged by objects e_{i+1} from the environment using the rules $r_{6,i}$.

The second process counts the number of vertices in the subset. It is performed using the rules $r_{7,ij}$ and $r_{8,ij}$, which interchange the objects B_i in the 2-cells by objects $\overline{B}_i$ (indicating this way that the corresponding vertex has been counted) and increase the counter d_j (the only purpose of the objects D_{ij} is to decrease the length of the rules). Note that this is a non-deterministic process, since the vertex "counted" in each step is chosen in a non-deterministic way. However, as the size of the subsets of vertices is upper bounded by n, after $2n$ steps the same configuration is always reached, so this stage is also confluent.

Note that for the counter d_j of a 2-cell to increase, it is necessary and sufficient that in that cell there exist objects B_i left. This means that at the end of the process explained in the previous paragraph, the only 2-cells that contain objects encoding subsets of size k are those containing the object d_{k+1}. At this moment, those cells also contain the counter e_{2n+1}, which then in two steps cause (using the rules $r_{9,j}$ and r_{10}, and the intermediate objects f_j for rules size reduction) the object d_{k+1} to be interchanged by objects l_1 and F_1 of the environment.

The *checking stage* starts now, but before we can check if any of the subsets of vertices of size k selected in the previous stage is an independent set of the graph, we need some preparation steps. First of all, the objects l_i will be used as a counter of the number of steps performed, controlled by rules $r_{11,i}$.

Simultaneously, the objects p and F_2 are traded against the object F_1 (by applying the rule r_{12}). In this way, the counters g and h appear (rule r_{13}) and they are duplicated (rules $r_{14,i}$ and $r_{15,i}$, respectively) till producing at least n copies of the object $g_{\lceil \lg\, n \rceil+1}$ and m copies of the object $h_{\lceil \lg\, m \rceil+1}$ respectively.

In the step $3n + \lceil \lg\, m \rceil + 7$, the rule $r_{16,ij}$ produces the trade of the objects $\overline{A}_{ij}$ and $h_{\lceil \lg\, m \rceil + 1}$ against the objects P_{ij}. On the other hand, in the step $3n + \lceil \lg\, n \rceil + 7$ the objects $\overline{B}_i$ and $g_{\lceil \lg\, n \rceil + 1}$ are traded against the objects B_{i1}. Along the following m steps, the rules $r_{18,ij}$ (i=1, ..., n and $j = 1, \ldots, m$) are applied and the produce the occurrence of m objects B'_i (one in each step) in the cells with label 2 which remain active after the pre-checking stage.

In the next step, the rules $r_{19,ij}$ are applied in the cells with label 2. Such rules trade the objects P_{ij} and B'_i against the objects P_j, and, in the next step, the objects P_j and B'_j are trade against an object β by means of the rules $r_{20,j}$.

Since m copies of the object B'_i are obtained, then the last step in which an object β can appear in a cell 2 is the step $3n + m + \lceil \lg\, n \rceil + 9$. In this way, the following step is the last one in which an object p can be sent to the environment (by applying the rule r_{21}, and by finishing the checking stage.

The *answer stage* starts when an object $l_{m+\lceil \lg\, n \rceil + 7}$ appears in a cell with label 2. Two possibilities must be considered.

If there exists an object p in a cell labelled by 2 when the checking stage is finished, then the subset encodes an independent set of size k of the graph $G = (V, E)$, and, by applying the ruler_{22}, at least one object T appears in the cell. Then, the rules r_{23}, r_{24} and r_{25} are applied and an object `yes` is sent to the environment. This ends the computation.

If no object p remains in any cell with label 2 after the checking stage, then the subset does not encode an independent set of the graph $G = (V, E)$ of size k and the rule r_{22} cannot be applied. Then, in the step $3n + m + \lceil \lg\, n \rceil + 12$, no object T arrives to the cell with label 1 and the rules r_{24} and r_{25} cannot be applied. In this way, the counter a reaches the object $a_{3n+m+\lceil \lg\, n \rceil + 14}$ which is traded together the object b against an object N by means of the rule r_{26}. Such object N is sent together with an object `no` to the environment in the step $3n + m + \lceil \lg\, n \rceil + 15$, and this ends the computation.

5 Conclusion and Future Work

As pointed above, the study of tractability in Membrane Computing opens a new perspective of the classical problem **P** vs. **NP** from a biologically inspired point of view. This research field is only a few years old and much work needs to be done in this line, not only in the theoretical results about tractability, but also in the development of new skills on the design of cellular solutions.

Following the ideas used in [2] (where an efficient solution of the 3–coloring problem was presented), an schema for solving **NP**–complete problems of graph theory has been inferred. It is used in this paper for presenting a first (linear–time) solution to the Independent Set problem.

More open questions in the framework of Membrane Computing related to tractability can be considered. For example, it is interesting to investigate the possibility to address efficient solutions to **NP**-complete problems in different frameworks allowing to produce an exponential number of cells in linear time (e.g. cell creation or separation).

It is also worth investigating a lower bound on the length of the symport/ antiport rules used by the system (in this paper, rules of length at most 3 are used). Another research problem is related to the number of objects in the environment. One could also consider a tissue-like P system where the environment always has a finite amount of objects.

Acknowledgment

The authors wish to acknowledge the support of the project TIN2006-13425 of the Ministerio de Educación y Ciencia of Spain, cofinanced by FEDER funds, and the support of the project of excellence TIC-581 of the Junta de Andalucía.

References

1. Díaz–Pernil, D.: Sistemas P de Tejido: Formalización y Eficiencia Computacional. PhD Thesis, University of Seville (2008)
2. Díaz–Pernil, D., Gutiérrez–Naranjo, M.A., Pérez–Jiménez, M.J., Riscos–Núñez, A.: A uniform family of tissue P system with cell division solving 3-COL in a linear time. Theoretical Computer Science 404, 76–87 (2008)
3. Díaz–Pernil, D., Gutiérrez–Naranjo, M.A., Pérez–Jiménez, M.J., Riscos–Núñez, A.: Solving Subset Sum in linear time by using tissue P systems with cell division. In: Mira, J., Álvarez, J.R. (eds.) IWINAC 2007. LNCS, vol. 4527, pp. 170–179. Springer, Heidelberg (2007)
4. Bernardini, F., Gheorghe, M.: Cell Communication in Tissue P Systems and Cell Division in Population P Systems. Soft Computing 9(9), 640–649 (2005)
5. Freund, R., Păun, G., Pérez-Jiménez, M.J.: Tissue P Systems with Channel States. Theoretical Computer Science 330, 101–116 (2005)
6. Krishna, S.N., Lakshmanan, K., Rama, R.: Tissue P Systems with Contextual and Rewriting Rules. In: Păun, G., Rozenberg, G., Salomaa, A., Zandron, C. (eds.) WMC 2002. LNCS, vol. 2597, pp. 339–351. Springer, Heidelberg (2003)
7. Martín Vide, C., Pazos, J., Păun, G., Rodríguez Patón, A.: A New Class of Symbolic Abstract Neural Nets: Tissue P Systems. In: H. Ibarra, O., Zhang, L. (eds.) COCOON 2002. LNCS, vol. 2387, pp. 290–299. Springer, Heidelberg (2002)
8. Martín Vide, C., Pazos, J., Păun, G., Rodríguez Patón, A.: Tissue P systems. Theoretical Computer Science 296, 295–326 (2003)
9. Pérez-Jiménez, M.J., Romero-Jiménez, A., Sancho-Caparrini, F.: A polynomial complexity class in P systems using membrane division. Journal of Automata, Languages and Combinatorics 11(4), 423–434 (2006)
10. Păun, G.: Computing with membranes. Journal of Computer and System Sciences 61(1), 108–143 (2000)
11. Păun, G.: Membrane Computing. An Introduction. Springer, Berlin (2002)
12. Păun, A., Păun, G.: The power of communication: P systems with symport/antiport. New Generation Computing 20(3), 295–305 (2002)
13. Păun, G., Pérez-Jiménez, M.J., Riscos-Núñez, A.: Tissue P System with cell division. International Journal of Computers, Communications & Control III (3), 295–303 (2008)
14. P systems web page, `http://ppage.psystems.eu/`